OCHD:Preserving Obliviousness Characteristic of Honeypot Database

S. K. Gupta† Renu Damor† Anand Gupta‡ Vikram Goyal†

† Indian Institute of Technology, Department of Computer Science & Engineering, New Delhi, India
‡ Netaji Subhas Institute of Technology, Division of COE, New Delhi, India
skg@cse.iitd.ernet.in, renu.damor@yahoo.com, nsit_anand@yahoo.com, vkgoyal@cse.iitd.ernet.in

Abstract

The objective of honeypot database (context honeypot) is to identify a potential privacy violator. A suspected user's interaction with the database is analyzed to determine suspicion. Such a system must satisfy certain characteristics. One of the such characteristics is obliviousness. The success of such a system depends on its remaining oblivious to the suspected user. In this demo paper, we describe the architecture and workflow of such a system, abbreviated as OCHD (**O**bliviousness **C**haracteristic of **H**oneypot **D**atabase).

1 Introduction

Honeypots are being used as security tools in network domain as early warning systems[1, 2, 3, 7]. Rakesh Agrawal et. al. [6] have proposed a framework of privacy policy for databases. From then onwards there has been a lot of research on security of privacy and prevention of unauthorized access to databases. Various functional applications such as hospital, university, passport, etc. gather personal information from the users. However, they require user's consent to opt-in / opt-out for disclosure of (parts of) data. This constitutes the privacy policy, which is enforced by the organization to prevent any unauthorized access to databases. However, even in a robust implementation, such a system can be compromised in different ways. Possibility of a user X masquerading as a user Y may exist. A suspected user could be prevented from accessing the database. A better approach would be to confirm the suspicion of a suspected user and then either (a) permit him/her to access the database (in case the suspicion is not well founded), or (b) deny access.

We address this issue by applying the concept of context honeypot. The suspected user is kept under intense observation to confirm or reject the suspicion [4]. Context honeypots are only meaningful for a specific malafide intension. The most important characteristic of context honeypot is obliviousness. This along with other characteristics are addressed by Gupta et. al. in the paper [5].

Obliviousness means that a suspected user, who has been put in honeypot should not in any way become aware of being put in honeypot. If he/she become aware that she/he is under observation, the whole exercise becomes self defeating. Thus, absence of this characteristic would make honeypot totally useless. In the context honeypot, the suspected user is allowed to interact so that his interaction details can be recorded for determination of suspicion in way to confirmation.

In the demonstration, we consider the example of *Disclosure of an actress Health information to News Media* as discussed in [5]. The reader may refer to the paper for details of the example. We consider the problem of disclosure of private information of a patient (an actress) by a suspected nurse. It is only possible when the nurse masquerades as a treating doctor (when robust privacy policy is in place). We address the issue of obliviousness in this demonstration. Here synthetic information is made available for the part which is not known to the suspected user.

The organization of the paper is as follows: Section 2 gives the architecture of OCHD, Section 3 discusses the workflow that describes techniques of obliviousness. Finally, Section 4 presents the demonstration and that completes the paper.

2 OCHD Architecture

The proposed architecture of OCHD (**O**bliviousness **C**haracteristic of **H**oneypot) is given in Figure 1. It comprises of following four Modules

1. User Interface
2. Masquerader Determinator
3. Query Manipulator
4. Synthetic Result Generator

These modules are described below :-

International Conference on Management of Data
COMAD 2006, Delhi, India, December 14–16, 2006

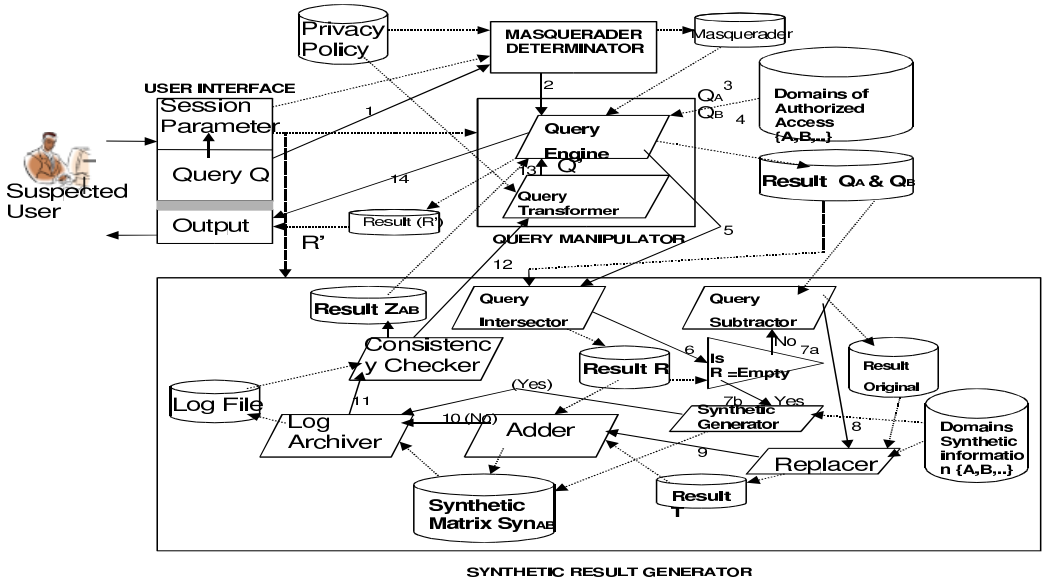

Figure 1: The Architecture of OCHD

2.1 User Interface

This module consists of three submodules and provides an interface between the user and the system.

1. **Session Parameter** Session parameters consist of IP address, day, duration, time, login/password etc of current session. Some of them have to be given by the user and the remaining parameters have to be generated by the system.

2. **Query** It accepts the Query Q posed by the user.

3. **Output** The Response result with reference to the query Q is given by the proposed system to the user.

2.2 Masquerader Determinator

Under the assumption that "User A is masquerading as user B", this module attempts to determine or make a knowledgeable guess for the identity of A (it may be noted that B is already known through session parameters). The reader is referred to Gupta et. al. [5] for details. Output of this module is fed to the query manipulator. For our purposes, we assume that this module has determined A.

2.3 Query Manipulator

It consists of two submodules:-

1. **Query Transformer** Inputs for query transformer are:- (a) User defined query Q and (b) output of the masquerader determinator. The task of query transformer is to generate a query Q' which will display that data which is asked by the user but modified as per privacy policy of the system. So, the output of query transformer is Q' which is directed to query engine for evaluation purpose.

2. **Query Engine** Inputs to the Query Engine are:- (a) either user defined query Q or the transformed query Q', and (b) output of the masquerader determinator. The output of query engine is the resultant query (either Q or Q').

2.4 Synthetic Result Generator

It consists of eight submodules:-

1. **Query Intersector** This submodule takes result of query Q w.r.t User A & User B. Its aim is to determine the information common to the masquerader and the masqueraded user. It will help in disclosing the original part of the information. The intersection of the query result is stored in *Result R*.

2. **Comparator(R=∅ ?)** This is a comparator box, whose result output is either True or False. The input to the box is from the input R, which is checked for empty set. If R=∅ is true then the control goes to Synthetic Generator module otherwise retrieves to Query Subtractor.

3. **Query Subtractor** This submodule retrieves the stored data from $Result\ of\ Q\{\ A,B\ \}$. Its aim is to find the information not known to the masquerader, so that, the information from synthetic table can be referred to accordingly. The output is the original part of $(A-B)$.

4. **Synthetic Generator** This submodule replaces in totality the requested view with the synthetic information.

5. **Replacer** This submodule with respect to the view $(A-B)$ replaces the information with synthetic part. This synthetic information will be generated randomly in a manner which will help in the confirmation/elimination of the suspicion. It will remain static throughout the interaction and will be different for each probable privacy violator.

6. **Adder** The objective of this submodule is the generation of combined view by fusing the outputs of *Result R* & *Result T*. This combined view is stored in Log Archiver and termed as Syn_{AB}.

7. **Log Archiver** The task of this submodule is to create log of each access and store all the details about each access. These details include Syn_{AB}, session parameters and both original and synthetic information displayed to the masquerader.

8. **Consistency Checker** Successful implementation of Obliviousness characteristic greatly depends upon the consistency of the shown data. Consistency checker takes care these issue. It uses log archiver to generate result Z_{AB}, so that consistency is not violated in reference to what has been shown to the masquerader earlier.

3 Workflow

In order to visualize the workflow of OCHD(**O**bliviousness **C**haracteristic of **H**oneypot **D**atabase) architecture, we refer to the example undertaken in [5], where malafide intension is to get details of a particular Actress using Login Id and password of Doctor, as the probable violator can be a nurse or a billing officer. Figure 1 gives the proposed architecture. The numbers on the solid arrows in Figure 1 indicate the sequence of processing of modules. The dotted lines indicate data sources. The workflow is described below.

1. The probable privacy violator opens login page of the hospital application and enters login id and password of the doctor. As login id and password are correct, authentication system permits the suspected user to access database.

2. Now the privacy violator (who is now in the role of doctor by masquerading) poses query Q regarding details about the actress. IP address being used by violator, access time and similar other details are recorded in the submodule Session Parameter. Q is now passed on to Query Manipulator for further steps.

3. In order to determine the identity of the probable privacy violator (nurse or billing officer), the module Masquerader Determinator is used.

4. The details of data which the authorized functionaries (Doctor, Nurse, Billing Officer) are supposed to know, are listed below and are stored in Domains of authorized access.

   Doctor → Doctor Id, Patient Id, Present and Past History of Illness, Patient age, Patient sex, Marital status of Patient , Nurse ID, Nurse Action, Nurse Report, Technician Id, Test Id, Test Report, Medicine Id, Patient Diagnose Over.

   Nurse → Nurse Id, Patient Id, Patient age, Patient sex, Nurse Action, Nurse Report, Medicine Id, Patient Diagnoses Over.

   Billing Officer → Billing Officer Id, Patient Id, Test Id, Doctor Id, Nurse Id, Medicine Id, Doctor rate, Nurse rate, test rate, medicine rate, Bill status.

5. Using data of Doctor and Nurse from storage of Domains of Authorized Access, value of R is determined using Query Intersector. R for this example is a set of values of the attributes common to both :- Nurse ID, Patient ID, Patient Age, Patient Sex, Nurse Action, Nurse Report, Medicine ID, Patient diagnose over.

6. Since R is a non empty set, the value of T is determined using Query Subtractor. Output of Query Subtractor is a set of values of these attributes :- DocId, Present and Past history of Illness, Marital Status, Techinician ID. Synthetic values for this set will be generated.

7. The output of Query Subtractor is replaced with synthetic information and then unified with Query Intersector using submodule Adder. This output is termed as Syn_{AB}.

8. The submodule Consistency Checker takes Syn_{AB} as input and verifies from Log Archiver for maintaining consistency of the part to be shown to the suspected user. The output after verification results in Z_{AB}.

9. The privacy policy is enforced to transform query Q into query Q'. This transformation is necessary to provide limited disclosure to the probable privacy violator. Query Q' is submitted to Query Engine, which evaluates the query Q' using the data Z_{AB} and the resultant view is stored inR' and shown to the suspected user against the query Q. The suspected user, if so desires may pose a fresh query Q and continue to remain in the proposed workflow.

10. The processes listed at paras 2 to 9 will continue till the suspected user logs out.

4 The Demonstration

In the proposed demonstration, we focus on the obliviousness characteristic of the context honeypot. This is based on the perceived knowledge of masquerader and the masqueraded identity. We have built a prototype of such a system for medical domain where the specific malafide intension is *Disclosure of an actress health information to news media*. If a suspected user becomes aware that he has been put in the honeypot then he can take remedial step to escape and that will defeat the purpose of the honeypot.

5 Acknowledgment

Our thanks to Surendra Tomar, Pratibha, Ashish, Sheetal Tewari, Yogesh and Navneet of Database Group in I.I.T Delhi for their invaluable suggestions. We wish to thank the N.T.R.O. and Ministry of Information Technology, Government of India for funding the project.

References

[1] Feng Zhang; Shijie Zhou; Zhiguang Qin; Jinde Liu. Honeypot: a supplemented active defense system for network security Network security Proceedings of Fourth International Conference on Parallel & Distributed Computing Applications and Technologies, 27-29 Aug. 2003,PDCAT 2003 pp. 231-235

[2] Scottberg, B.; Yurcik, W.; Doss, D. Internet honeypots: protection or entrapment? International Symposium on Technology and Society, ISTAS'02, 6-8 June 2002, pp. 387 - 391

[3] Spitzner, L. Honeypots: Catching the insider threat Ninteenth Annual Computer Security Applications Conference, ACSAC 2003, pp. 170 - 179

[4] Gupta, S.K.; Gupta A.; Goyal, V.; Sabharwal, S.;Patra, B.; Damor, R. Context honeypot : A framework for anticipatory privacy violation Submitted in Fourth International Conference on Bridging the Digital Divide, AACC 2006, Nepal, December 2006

[5] Gupta, S.K.; Gupta A.; Damor, R.; Goyal, V.; Sabharwal, S. Preserving Obliviousness Characteristic of Honeypot database 13th International Conference on Data Management, COMAD 2006, New Delhi, 14-16 December 2006

[6] Agrawal, R; Kiernan, J.; Srikant, R.; Yirong Xu. Hippocratic databases Proceedings of the Twenty-eight International Conference on Very Large Bases, 2002, pp. 143-54

[7] Provos, N. Honeyd: A Virtual Honeypot Framework Freely Available software; downloaded from : http://www.citi.umich.edu/u/provos/honeyd/